中国•珠海 全志科技股份有限公司 All Winner Tech. CO. LTD	文档编号				
	文档名称	固件打包流程分析			
	版　　本	V1.0	密　　级	1	共 1 页

固件打包流程分析

（内部公开）

文档作者	zhouhuacai			创建日期	2016-02-01

拟　　制	zhouhuacai	日　　期	2016-02-01
审　　核		日　　期	
批　　准		日　　期	

分发部门							
[　]SW	[　]SD	[　]AL	[　]CS	[　]TST	[　]MKT		

版本历史

	修改人	时间	备注
V1.0	zhouhuacai	2016-02-01	初始版本

目 录

1. 概述

1.1. 编写目的

介绍固件打包流程，为想了解固件打包流程的人员提供概要信息。

1.2. 适用范围

适应平台为 boot2.0 平台。

1.3. 相关人员

公司开发人员及关心固件打包的其它人员。

2. 固件打包介绍

固件打包是指将我们编译出来的 bootloader，内核，和根文件系统一起写到一个镜像文件中，这个镜像文件也叫固件。然后可以将这个镜像写到 nand flash 或是 sd 卡上，从而启动系统。

3. 打包脚本分析

3.1. 脚本的调用流程

编译完成后便可打包,打包方法,在 lichee 目录下运行:

```
#./build.sh pack
```

最终打包出来的固件放在目录 lichee/tools/pack/下.

打开 build.sh 脚本,发现只有一行命令:

```
buildroot/scripts/mkcommon.sh $@
```

这一行命令表示运行 buildroot/scripts/目录下的 mkcommon.sh 脚本,同时参数"pack"传给 mkcommon.sh.即相当于运行:

```
#buildroot/scripts/mkcommon.sh pack
```

打开 mkcommon.sh 脚本,根据传入的参数 pack,脚本里执行:

```
elif [ "x$1" = "xpack" ] ; then
    init_defconf
    mkpack
    exit $?
```

也就是在该脚本里调用了 init_defconf 和 mkpack 函数.这两个函数在另一脚本 mkcmd.sh 实现.

init_defconf 函数主要是根据用户选择的 chip,platform,board 等信息,配置相应的目录,以使用对应的目录文件.

mkpack 函数主要是运行 pack 进行打包.

```
 ./pack -c ${LICHEE_CHIP} -p ${LICHEE_PLATFORM} -b ${LICHEE_BOARD} -k ${LICHEE_KERN_VER} $@)
```

例如如使用配置 chip 为 sun8iw11p1,platform 为 linux,board 为 v40_perf1_v1_0 则相当于 ./pack -c "sun8iw11p1" -p "linux" -b "v40_perf1_v1_0" -k "linux-3.10"

pack 其实也是一个脚本文件,目前打包脚本分为 5 个阶段，分别是:

```
do_prepare
do_ini_to_dts
do_common
do_pack_${PACK_PLATFORM}
do_finish
```

3.2. 打包的各阶段分析

3.2.1. do_prepare 阶段

此阶段完成文件拷贝动作。打包时需要拷贝若干文件到 tools/pack/out 目录下，目前脚本对其进行了分类，分别是 tools_file_list，configs_file_list，boot_resource_list 和 boot_file_list，如有新增文件,可以归入其中一类或者创建新类,后续打包会使用到这些文件.

具体实现分析如下:

```
function do_prepare()
{
    ......

    #拷贝 tools_file_list 类文件到 tools/pack/out 目录下
printf "copying tools file\n"
for file in ${tools_file_list[@]} ; do
    cp -f $file out/ 2> /dev/null
done

    #拷贝 configs_file_list 类文件到 tools/pack/out 目录下
printf "copying configs file\n"
for file in ${configs_file_list[@]} ; do
    cp -f $file out/ 2> /dev/null
done

    #拷贝 boot_resource_list 类文件到 tools/pack/out 目录下
printf "copying boot resource\n"
for file in ${boot_resource_list[@]} ; do
    cp -rf `echo $file | awk -F: '{print $1}'` \
    `echo $file | awk -F: '{print $2}'` 2>/dev/null
done

    #拷贝 boot_file_list 类文件到 tools/pack/out 目录下
printf "copying boot file\n"
for file in ${boot_file_list[@]} ; do
    cp -f `echo $file | awk -F: '{print $1}'` \
    `echo $file | awk -F: '{print $2}'` 2>/dev/null
done

......

}
```

3.2.2. do_ini_to_dts 阶段

在 linux-3.10,引入了 linux 设备树的概念.此阶段编译生成描述设备树的 sunxi.dtb 文件.该文件在 linux 内核启动过程中会被解析，根据该文件中设备列表进行加载各个外设的设备驱动模块.

具体实现分析如下:

```
function do_ini_to_dts()
```

```
{
    if [ "x${PACK_KERN}" != "xlinux-3.10" ] ; then
        return
    fi

......
    #如果是 linux-3.10 内核,编译生成.dbt 文件
$DTC_COMPILER -O dtb -o ${LICHEE_OUT}/sunxi.dtb \
        -b 0            \
        -i $DTC_SRC_PATH    \
        -F $DTC_INI_FILE    \
        -d $DTC_DEP_FILE $DTC_SRC_FILE
    if [ $? -ne 0 ]; then
        pack_error "Conver script to dts failed"
        exit 1
    fi

    printf "Conver script to dts ok.\n"

......
```

3.2.3. do_common 阶段

此阶段完成所有系统平台通用的文件解析，分区打包.

具体实现分析如下:

```
function do_common()
{
    cd out/

  busybox unix2dos sys_config.fex
  busybox unix2dos sys_partition.fex

    #使用 script 程序解析文本文件 sys_config.fex 和 sys_partition.fex
    #生成相应的二进制文件 sys_config.bin 和 sys_partition.bin 便于程序解析
    script   sys_config.fex > /dev/null
    script   sys_partition.fex > /dev/null
    cp -f    sys_config.bin config.fex

  .......

    if [ -f "${LICHEE_OUT}/sunxi.dtb" ]; then
    #更新 u-boot.fex,在 u-boot.fex 指定的位置加入 sunxi.fex 的配置信息
        cp ${LICHEE_OUT}/sunxi.dtb sunxi.fex
```

```
        update_uboot_fdt u-boot.fex sunxi.fex u-boot.fex >/dev/null
    fi

    ......

    #根据 sys_config.bin 参数,取出 DRAM,UART 等参数更新 boot0 头部参数
    update_boot0 boot0_nand.fex sys_config.bin NAND > /dev/null
    update_boot0 boot0_sdcard.fex  sys_config.bin SDMMC_CARD > /dev/null
    #根据 sys_config.bin 参数设置,更新 uboot 头部参数
   update_uboot u-boot.fex  sys_config.bin > /dev/null
    #根据 sys_config.bin 参数设置,更新 fes1.fex 参数
    update_fes1  fes1.fex  sys_config.bin > /dev/null
    #制作启动过程相关资源的分区镜像
    fsbuild  boot-resource.ini  split_xxxx.fex > /dev/null

    #根据配置,生成 uboot 基本配置二进制文件 env.fex
u_boot_env_gen env.cfg env.fex > /dev/null

    if [ -f boot_package.cfg ]; then
        echo "pack boot package"
        busybox unix2dos boot_package.cfg
        dragonsecboot -pack boot_package.cfg
    fi

}
```

3.2.4. do_pack_${PACK_PLATFORM}阶段

此阶段完成当前系统平台特有的工作.

例如是 linux 系统平台,则执行 do_pack_linux.主要对内核文件,文件系统等进行软链接.

```
function do_pack_linux()
{
    if [ "x${PACK_KERN}" != "xlinux-3.10" ] ; then
        return
fi

printf "packing for linux\n"
    #软链接 vmlinux.fex,boot.fex,rootfs.fex
ln -s ${LICHEE_OUT}/vmlinux.tar.bz2 vmlinux.fex
ln -s ${LICHEE_OUT}/boot.img  boot.fex
ln -s ${LICHEE_OUT}/rootfs.ext4  rootfs.fex
......
```

3.2.5. do_finish 阶段

此阶段根据指定的固件成员完成打包.

具体实现分析如下：

```
function do_finish()
{
......

    #生成分区结构文件 sunxi_mbr.fex 及分区下载文件列表文件 dlinfo.fex
update_mbr                sys_partition.bin 4 > /dev/null
    #根据所列的成员文件及分区信息,组合完成打包
dragon image.cfg       sys_partition.fex

......

}
```

4. 固件组成成员分析

固件包本质是由一系列的文件组成，类似于一个压缩包，把多个文件压缩成了一个固件包。这里，通过一个描述性的配置文件，把需要添加到固件包的文件枚举出来。然后，打包过程就读取这个配置文件，生成了最终的固件包。

用文本方式，打开\tools\pack\out\image.cfg 文件，可以看到大致如下的内容：

```
[DIR_DEF]
INPUT_DIR = "../"

[FILELIST]
    {filename = "sys_config.fex",       maintype = ITEM_COMMON,             subtype =
"SYS_CONFIG100000",},
    {filename = "config.fex",           maintype = ITEM_COMMON,             subtype =
"SYS_CONFIG_BIN00",},
    {filename = "split_xxxx.fex",       maintype = ITEM_COMMON,             subtype =
"SPLIT_0000000000",},
    {filename = "sys_partition.fex",maintype = ITEM_COMMON,                 subtype =
"SYS_CONFIG000000",},
    {filename = "sunxi.fex",            maintype = ITEM_COMMON,             subtype =
"DTB_CONFIG000000",},

    {filename = "boot0_nand.fex",       maintype = ITEM_BOOT,               subtype =
"BOOT0_0000000000",},
    {filename = "boot0_sdcard.fex", maintype = "12345678",                  subtype =
"1234567890BOOT_0",},
    {filename = "u-boot.fex",     maintype = "12345678",                    subtype =
"UBOOT_0000000000",},
    {filename = "toc1.fex",       maintype = "12345678",                    subtype =
"TOC1_00000000000",},
    {filename = "toc0.fex",       maintype = "12345678",                    subtype =
"TOC0_00000000000",},
    {filename = "fes1.fex",             maintype = ITEM_FES,                subtype =
"FES_1-0000000000",},
    {filename = "boot_package.fex", maintype = "12345678",                  subtype =
"BOOTPKG-00000000",},

    {filename = "full_img.fex",         maintype = "12345678",              subtype =
"FULLIMG_00000000",},

    {filename = "usbtool.fex",          maintype = "PXTOOLSB",              subtype =
"xxxxxxxxxxxxxxxx",},
    {filename = "aultools.fex",         maintype = "UPFLYTLS",              subtype =
```

```
"xxxxxxxxxxxxxxxx",},
    {filename = "aultls32.fex",          maintype = "UPFLTL32",          subtype =
"xxxxxxxxxxxxxxxx",},

    {filename = "cardtool.fex",          maintype = "12345678",          subtype =
"1234567890cardtl",},
    {filename = "cardscript.fex",        maintype = "12345678",          subtype =
"1234567890script",},

    {filename = "sunxi_mbr.fex",         maintype = "12345678",          subtype =
"1234567890___MBR",},
    {filename = "dlinfo.fex",            maintype = "12345678",          subtype =
"1234567890DLINFO",},
    {filename = "arisc.fex",             maintype = "12345678",          subtype =
"1234567890ARISC" ,},

[IMAGE_CFG]
version = 0x100234
pid = 0x00001234
vid = 0x00008743
hardwareid = 0x100
firmwareid = 0x100
bootromconfig = "bootrom_071203_00001234.cfg"
rootfsconfig = "rootfs.cfg"
;;imagename = "ePDKv100_nand.img"
filelist = FILELIST
;imagename = ..\sun4i_test_evb.img
encrypt = 0
```

该文件项的格式：

```
filename= name,   maintype=ITEM_ROOTFSFAT16,   subtype = user_define
```

当用户需要添加文件的时候，按照同样的格式，把自己需要的文件写到脚本文件中即可。

- **filename：打包文件**

  是指文件的全路径。可以使用相对路径，如上述文件中，就使用了相对路径。

- **maintype：打包格式**

  表明文件的格式类型,该文件有此类型定义的列表.

- **subtype：自定义名称**

  用户自己定义的名称，使用数字和英文字符(区分大小写)，最大长度必须为 16 字节.

只按照上述规则书写，并放到文件的[FILELIST]之后，等到打包的时候就会自动把文件添加到固件包中。

下表描述 image.cfg 文件中的各固件成员的作用.

固件成员	成员作用
sys_config.fex	从具体的 board 配置目录拷贝而来,此文件描述了一些硬件相关的配置,用户可以修改该配置.如果不关心某项配置,则可以直接

	删除该项,此时该项使用默认值.
config.fex	硬件相关的配置信息的二进制文件,用于程序解析.
split_xxxx.fex	作为 fsbuild 的其中一个输入参数
sys_partition.fex	规划分区文件，指明存储设备上的分区个数，并由用户定义分区属性。当烧写固件包后，存储设备上就会存在这样由用户定义的分区。
sunxi.fex	描述设备树的配置信息,内核会将这些资源展开相应的设备。
boot0_nand.fex	boot0 编译生成的 nand 启动目标代码,在 SRAM 中运行,主要作用是初始化 DRAM,并从外部存储器 nand 中加载 UBOOT。对 UBOOT 做效验并跳转到 UBOOT 执行。
boot0_sdcard.fex	boot0 编译生成的 sdcard 启动目标代码,在 SRAM 中运行,主要作用是初始化 DRAM,并从外部存储器 sdcard 中加载 UBOOT。对 UBOOT 做效验并跳转到 UBOOT 执行。
u-boot.fex	u-boot 编译生成的目标代码,主要作用是初始化时钟设置,电源管理,卡量产,USB 烧写等,最后加载内核
toc1.fex	用于 secboot
toc0.fex	用于 secboot
fes1.fex	用于初始化 DRAM，并返回初始化的结果。在小机进行 USB 量产或升级时，需要先运行这段代码。
boot_package.fex	对指定文件进行打包,目前只包含 uboot,便于扩展
full_img.fex	用于小容量的外部存储器如 nor flash,此时没有分区概念.如果是不是 nor flash 启动,这个文件会置空
usbtool.fex	Usb 烧写工具插件,处理 USB 烧写的整个过程,适用于 windows 系统
aultools.fex	usb 烧写工具插件,处理 USB 烧写的整个过程,适用于 linux 64 位系统
aultls32.fex	usb 烧写工具插件,处理 USB 烧写的整个过程,适用于 linux 32 位系统
cardtool.fex	Card 烧写工具.处理 card 烧写的整个过程
cardscript.fex	指定 Card 烧写的各分区文件
sunxi_mbr.fex	分区主引导记录
dlinfo.fex	指定分区 download 的文件
arisc.fex	小 CPU 的一段可执行代码,用于管理 standby,电源管理等

除 image.cfg 文件所列的文件,固件还包含了 sys_partition.fex 所列的分区的文件.
用文本文件打开 sys_partition.fex,可以看到大致如下的内容:

```
[partition_start]

[partition]
    name         = boot-resource
    size         = 65536
    downloadfile = "boot-resource.fex"
    user_type    = 0x8000
```

```
[partition]
    name         = env
    size         = 32768
    downloadfile = "env.fex"
     user_type    = 0x8000

[partition]
    name         = boot
    size         = 131072
    downloadfile = "boot.fex"
     user_type    = 0x8000

[partition]
    name         = rootfs
    size         = 1048576
    downloadfile = "rootfs.fex"
     user_type    = 0x8000
```

这是一个规划磁盘分区的文件.一个分区的属性，有如下几项：

- 分区名称
- 分区的大小
- 下载的文件
- 分区的用户属性

以下是文件中所描述的一个分区的属性：

- **name：分区名称**

分区名称由用户自定义。当用户在定义一个分区的时候，可以把这里改成自己希望的字符串,但是长度不能超过 16 个字节

- **size:分区的大小**

定义该分区的大小,以扇区的单位.

- **downloadfile:下载的文件**

下载文件的路径和名称，可以使用相对路径，相对是指相对于 image.cfg 文件所在分区。也可以使用绝对路径.

- **user_type：分区的用户属性**

提供给操作系统使用的属性。目前，每个操作系统在读取分区的时候，会根据用户属性来判断当前分区是不是属于自己的然后才进行操作。这样设计的目的是为了避免在多系统同时存在的时候，A 操作系统把 B 操作系统的系统分区进行了不应该的读写操作，导致 B 操作系统无法正常工作。

下表描述了 sys_partition.fex 文件指定的分区里的文件.

固件成员	成员作用
boot-resource.fex	启动过程用到的资源,包括 logo,图片,字库等
env.fex	u-boot 的基本配置文件
boot.fex	Buildroot 生成的 boot.img 的软链接，它由 kernel 与 ramdisk 组成

rootfs.fex	Buildroot 生成的 rootfs.ext4 的软链接,根文件系统